Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 794 – 802

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

Preliminary evaluation of a problem-posing method in programming classes

Hisayoshi Kunimune[a,*], Masaaki Niimura[b]

[a]*Faculty of Engineering, Shinshu University, 4–17–1 Wakasato, Nagano, 3808553 Japan.*
[b]*Division of Science and Technology, Shinshu University, 4–17–1 Wakasato, Nagano, 3808553 Japan.*

Abstract

We propose a method of problem-posing for improving students' problem solving ability in programming and develop a system for supporting to operate the method in actual classes. In the proposed method students pose problems by modifying examples provided by the teacher, and they evaluate problems posed by other students on the basis of the diversity. This paper describes the overview of the proposed method and the system, and the evaluation of the proposed method by introducing it into fundamental programming class.

Peer-review under responsibility of KES International.

Keywords: problem-posing; programming class; problem solving ability; diversity; peer-review

1. Introduction

The authors teach a C language programming class for computer science course freshmen at a university in Japan. In this class, the students learn the concept, grammar and technical details of the C language programming, and they are also tacitly required to acquire *algorithmic thinking* through programming exercises. Algorithmic thinking involves the gradual resolution of a given problem into procedures, which can be implemented in specified programming paradigms. However, we find that the number of students who do not consider resolving procedures is increasing[1].

Many researchers have studied programming languages and environments for novices since the early 1960s[2,3,4]. These languages and environments are designed to help understanding the concepts, grammar and technical details of programming by using graphical user interface, easier grammar and so on. However, it is difficult to develop algorithmic thinking. On the other hand, some previous studies have discussed the effect of problem-posing e.g.

* Corresponding author. Tel.: +81-26-269-5502 ; fax: +81-26-269-5502.
E-mail address: kunimune@cs.shinshu-u.ac.jp

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.162

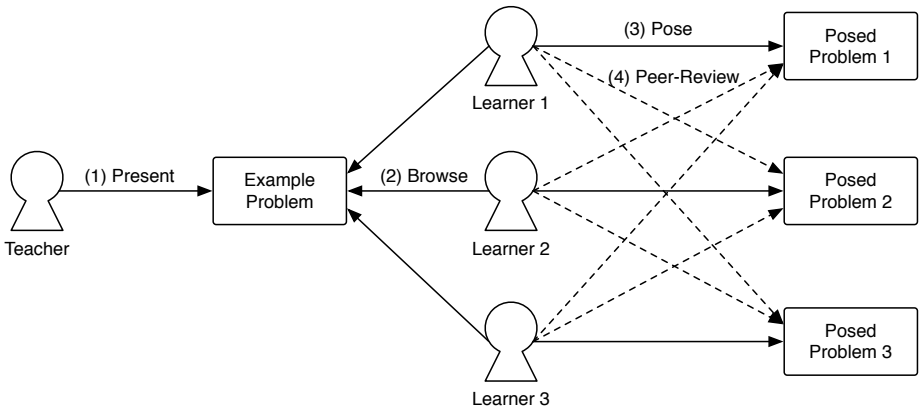

Fig. 1: Overview of proposed method.

enhancing problem-solving ability, improving understanding of solution, and promoting independence in learning arithmetic and mathematics[5,6]. Problem-posing is the type of exercises to make problems.

However, introducing problem-posing into an actual class might occur some issues. Hirai et al. introduced problem-posing in outside of school hours, and there were students who could not generate problems[7]. They pointed out that this issue has roots in lack of the student's understanding about learning contents, and it is the problems of the learning method. Similarly, Yu et al. emitted that it is important to find ways to assist students with insufficient knowledge[8]. On the other hand, generating diverse problems is very important in problem-posing[9]. Though students generate problems like those presented in textbooks or classes[10], it is difficult for them to generate diverse problems. Therefore, supporting students to generate diverse problems and reduce the load of problem-posing are important.

Renkl pointed out that mathematics, physics and programming are well-structured domains[11]. In the process of problem solving in these domains, learners consider vague solution and gradually resolve the solution into specific procedures with equations or programming languages. We thus develop a system, which supports to conduct problem-posing exercises in actual classes, and introduce problem-posing exercises into a programming class to develop students' ability of problem solving in programming[12].

This paper describes the overview of the method and the support system for introducing problem-posing exercises in programming classes and its evaluation.

2. Overview of proposed method and support system

The objective of this work is introducing problem-posing into a programming class to develop students' ability in programming. We only focus on posing word-problems in this work. A word-problem consists of a problem statement and its answer. The problem statement shows the students the situation and specification of the problem. The answer should be described in a programming language and fulfills the specification mentioned in the problem statement.

2.1. Proposed method

To achieve the objective, we proposed a method for problem-posing and developed a system for assisting to operate the method. Figure 1 illustrates the overview of proposed method. In this method, a teacher (1) presents an example problem for students to make posing problem easier. The students (2) browse the example, (3) pose a new problem by modifying the problem statement or the answer of the example, and (4) review posed problems among students (peer-reviewing) based on the evaluation viewpoints shown by the teacher before the exercise. Figure 2 shows the flow of reviewing by students.

The viewpoints of evaluation for peer-reviewing consist of the diversity and correctness of posed problem. Firstly, reviewers (students) evaluate the diversity of posed problem. If the problem is not diverse, and they describe some

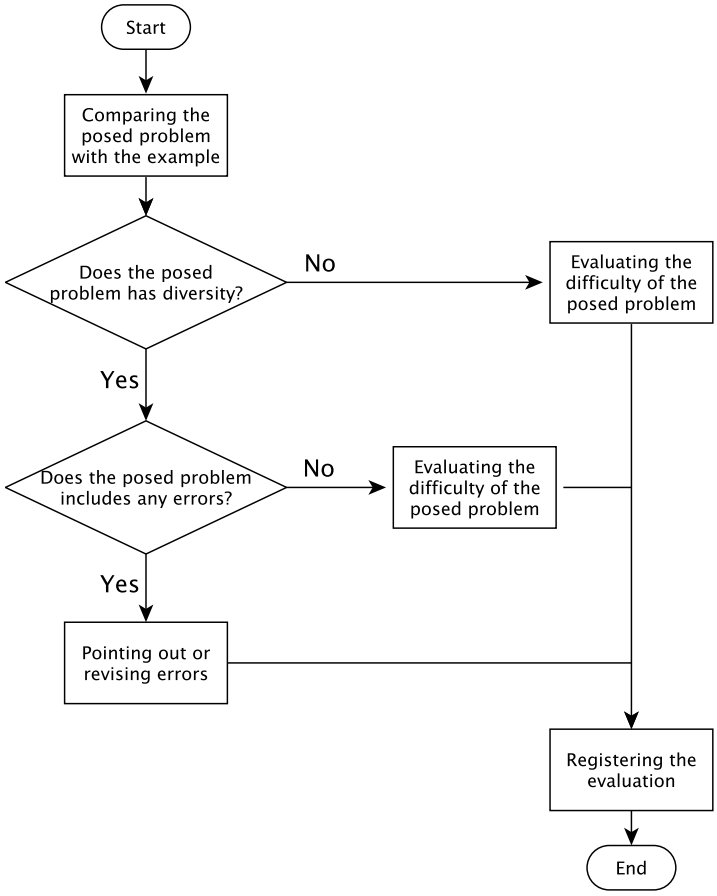

Fig. 2: Flow of reviewing.

ideas to make the problem diverse. In another case, they evaluate the correctness of the problem. If the problem has some errors, and they point out or revise the errors. In another case, they evaluate the difficulty of the problem.

Kojima et al. proposed the diversity of posed problems. It consists of two attributes: situation and solution[6]. We also used these attributes to evaluate the diversity of posed problems. A problem statement presents the situation of the problem, and its answer shows the solution of it.

In our method, reviewers evaluate the situation of the posed problem from verbs used in its problem statement. For example, "Tom walked 50 meters in 100 strides." and "Tom bought 50 pins for 100 yens." are in different situation because the verbs in these sentences are different. However, "Tom swam 50 meters in 100 seconds." is in the same situation of the first sentence because the verbs "walk" and "swim" represent the situation of moving. Problem statements, which are different from the example problem in expressions or verbs indicating the same situation, can be posed without considering other applicable situations of the solution of the example. Therefore, we defined that such differences do not bring diversity to posed problems. If there are some additions, modifications or deletions in the process of program (solution) of the posed problems from the example, and the problems are diverse in their solution. Programs, which are modified only in parameters in expressions, are not diverse in their solution because modifying these parameters does not need to consider the detail of programs.

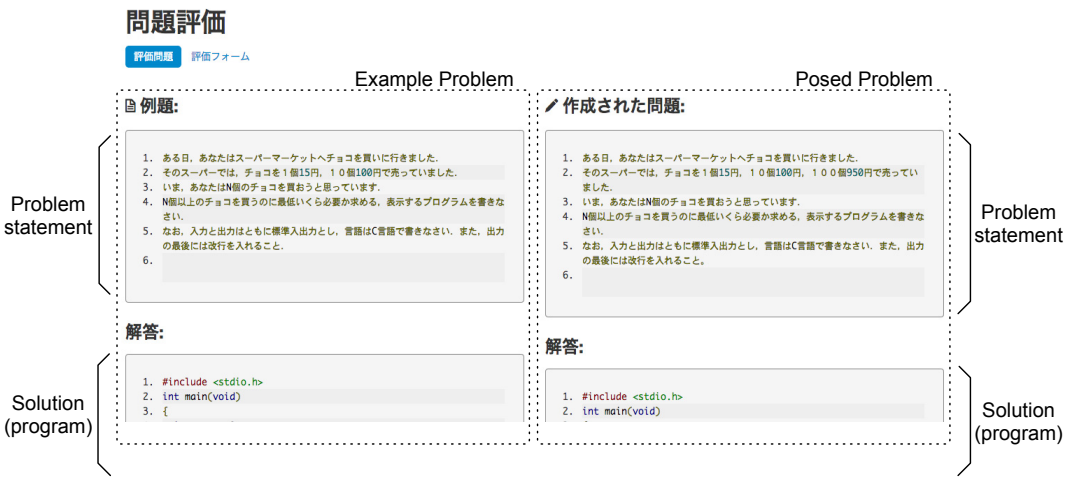

Fig. 3: Comparing the example (left side) with posed (right side) problems.

2.2. Support system

We developed a web-based system to support conducting the proposed method. A teacher (1) registers an example problem on the system. Then students (2) browse it, (3) pose and register a problem and (4) review problems posed by others on the system.

When a student registers a posed problem, the system assigns the problem to other students according to the reviewer assigning algorithm proposed by Fujihara et al.[13]. This algorithm avoids imbalance of the number of assigned problems between students. A student reviews assigned problems by comparing the example with each problem. Figure 3 shows the screen for reviewing a posed problem in the proposed system. The system shows these problems side by side to help reviewers to compare them.

The system changes input forms according to the student's evaluations to clarify the flow of evaluations. Figure 4 shows the difference between evaluation forms shown by the system according to the evaluation of the diversity. For example, if a student evaluates a problem as diverse, and the system shows the radio buttons for judging the correctness of the problem as shown in Figure 4a. Also if the student evaluates the problem has some errors, and the system shows the text area for writing ideas to revise them. If the student evaluates that the problem does not include any errors, and the system shows the radio buttons for judging the difficulty of the problem. The student evaluates that the problem is not diverse, and the system only shows the text area for writing ideas to make the problem diverse as shown in Figure 4b.

3. Experiment

We conducted an experiment to confirm the effect of the proposed method from May 23 to July 4, 2013 in the supplementary lectures of a fundamental C language class. Almost all of the participants of the class were sophomores in a computer science course, who failed to earn the credit of this class last year. The experimental procedure consisted of three parts: pre-test, tasks and post-test.

3.1. Pre-test

We conducted a pre-test in May 23 to measure the programming ability of the participants before introducing the proposed method. The pre-test consisted of two reading problems #1 and #2 (including six subquestions in each problem) and three writing problems #3, #4 and #5. The reading problems required the participants to answer the execution result of a program to confirm their understanding about the behavior of programs. The writing problems

(a) Evaluation form for problems with diversity.

(b) Evaluation form for problems without diversity.

Fig. 4: Evaluation forms in the system.

required the participants to write a program, which fulfills the specification mentioned in the problem statement, to confirm their problem-solving ability in programming. Each writing problem required writing a program including only conditional branches (problem #3), only repetitions (#4) and both of them (#5), respectively. Figures 5a and 5b show the example of reading and writing problems in the pre-test.

3.2. Tasks

We introduced problem-posing exercises into the supplementary lecture for five weeks from May 30 to June 27. The participants took the lectures for 90 minutes once a week.

In the first class in May 30, we instructed the participants about the problem-posing, the viewpoints of evaluation and the usage of the system. The participants browsed an example problem and posed a problem after that. In the second class in June 6, they reviewed three problems posed by other participants in last class. They also posed a problem with new example.

In the third class in June 13, the teacher instructed the viewpoints of evaluation again because there are some participants who did not understand the viewpoints. The participants reviewed three problems posed by others in last class, and they posed another problem with the example presented in last class. In the fourth class in June 20, they reviewed three problems posed by others in last class. They also posed a problem with new example.

In the fifth class in June 27, they reviewed three problems posed by others in last class, posed a problem with new example, and reviewed three problems posed in this class. After that, we conducted questionnaire survey to collect their opinions with open-ended questions.

3.3. Post-test

We conducted a post-test in July 4 to measure the programming ability of the participants after introducing the proposed method. The composition of the test was the same as it of the pre-test; however, the problems used in these tests are different.

4. Experimental result

4.1. Result of the pre- and post-tests

The number of the participants who took both the pre- and post-tests are 21. In this section, we mention the experimental result about them.

Figure 6 shows the histogram of the numbers of correct answers in reading problems (12 subquestions). All of the participants correctly answered more than half of questions; thus, we find that they understood basic grammar of the C language and the behavior of C language programs.

Answer the character printed by below program.

```
#include <stdio.h>

int main(void) {
    int i;
    scanf("%d", &i);
    if ( (1) ) {
        printf("a");
    else {
        printf("b");
    }
    return 0;
}
```

When the expression (1) is "`i < 0`",
input "`10`", and
_____ is printed.
input "`0`", and
_____ is printed.
input "`-10`", and
_____ is printed.

When the expression (1) is "`i <= 0`",
input "`10`", and
_____ is printed.
input "`0`", and
_____ is printed.
input "`-10`", and
_____ is printed.

(a) Example of reading problem.

```
#include <stdio.h>

int main(void) {
    int n;
    scanf("%d", &n);

    (1)

    return 0;
}
```

Complete (1) of this program to fulfill following specification.
- get an integer n (n is more than zero)
- print the digit number of n

(b) Example of writing problem.

Fig. 5: Examples of problems in the pre-test.

Table 1: Numbers of participants who answered writing problems correctly.

	Problem #3	Problem #4	Problem #5
Pre-test	18	15	2
Post-test	21	15	18

Table 1 shows the numbers of the participants who correctly solved each writing problem except grammatical errors. The result of the problem #5 between the tests shows a large difference. Figures 7a and 7b show the problem statements of both problems.

We categorize the errors in the incorrect answers of the problem #5 as follows:

- Sequence of conditional branches (E1)
- Omission of conditions (E2)
- Incorrect conditions (E3)
- Range of repetition (E4)
- Others (E5)

Table 2 shows the numbers of the incorrect answers of the problem #5, which are categorized into E1 to E5.

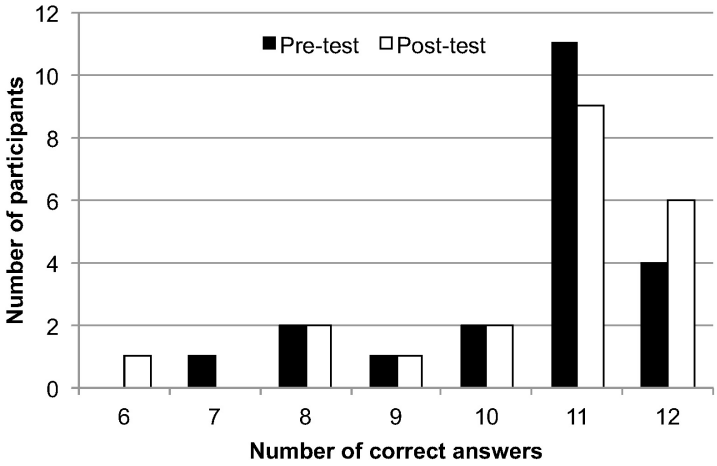

Fig. 6: Histogram of correct answers in reading problems.

```
#include <stdio.h>

int main(void) {
    int i;
    scanf("%d", &i);

    (1)

    return 0;
}
```

Complete (1) of this program to fulfill following specification.
- print integers from 1 to 30
- print "fizz" instead of 3's multiples
- print "buzz" instead of 5's multiples
- print "fizzbuzz" instead of 3 and 5's multiples.

(a) Problem #5 in pre-test

```
#include <stdio.h>

int main(void) {
    int n;
    scanf("%d", &n);

    (1)

    return 0;
}
```

Complete (1) of this program to fulfill following specification.
- get an integer n (n is more than 0)
- print all divisors of n

(b) Problem #5 in post-test

Fig. 7: Problem #5 in pre- and post-test.

The correct answer of the problem #5 in the pre-test includes more than one conditional branches. It also needs to choose proper output in repetition. On the other hand, the correct answer of the problem #5 in the post-test includes only one branch. Therefore, E1 and E2 are the errors peculiar to the problem #5 in the pre-test, and we omitted these errors from analysis. E3, E4 and E5 are possible in the both problems, although the numbers of participants, who made E3 and E4 in the post-test are less than one in the pre-test.

We also analyzed the diversity of the problems posed by the participants in the exercises. At first, we analyzed the correspondence between the evaluations of the diversity by the teacher and the participants to confirm whether the participants understand the diversity of problems. Table 3 shows the degree of correspondence. The degrees of the

Table 2: Category of errors in incorrect answers of problem #5.

	E1	E2	E3	E4	E5
Pre-test	6	6	4	3	0
Post-test	0	0	0	1	2

Table 3: Correspondence degree between evaluations by the teacher and the participants.

Exercise #1	Exercise #2	Exercise #3	Exercise #4	Exercise #5
0.89	0.86	0.95	1.0	1.0

Table 4: Rate of diverse problems posed by the participants in each category.

	NG/OK	NG/NG	OK/NG	OK/OK
E3	0.82	–	–	0.79
E4	0.86	–	0.50	0.78
E5	–	–	0.33	0.85

(–: no participants in the category.)

exercise # 1 and 2 are less than those of other exercises because the teacher explained the viewpoints of evaluation again at the beginning of the third lecture. Thus, we narrow the subject of the analysis to the exercises # 3 to 5.

We categorized the participants according to the answers of the problem #5 in the pre- and post-tests as follows:

- made an error in pre-test / did not make error in post-test (NG/OK)
- made an error in pre- and post-tests (NG/NG)
- did not make error in pre-test / made an error in post-test (OK/NG)
- did not make error in pre- and post-tests (OK/OK)

Table 4 shows the rate of diverse problems posed by the participants in the exercises in each category. According to this result, the NG/OK and OK/OK participants posed more diverse problems than the OK/NG participants did. This result suggests the effect of the proposed method for developing problem solving ability in programming. However, there are no participants in some categories, and we should conduct experiments to collect data and analyze these results.

4.2. Result of questionnaire survey

Some of the participants expressed negative opinions in the survey as follows:

- These exercises only needed known things to me, so I did not acquire new knowledge.
- I did not understand the intention of these exercises.
- I did not have enough knowledge and skill of programming to pose problems.

We found that some participants do not understand the intention of introducing problem-posing into the class according these opinions, and there are some participants who cannot pose problems based on examples.

According to the experimental result, we found the effect of the proposed method in an actual class and that some participants did not understand the intention of the problem-posing exercises. We guess the causes of this situation that (1) our explanation was insufficient, (2) they have mainly took knowledge transmission lecture. Thus, we should improve our instruction to meet the participants' characteristics.

The system implements the reviewer assigning algorithm to avoid imbalance of the number of assigned problems between students. However, this algorithm does not treat the quality of assigned problems. The quality of the problems

assigned to a student may affect the development of the student's programming or problem-posing ability. We, thus, should improve the assigning algorithm to consider the quality of the problems.

Furthermore, we should consider supporting students, who cannot pose problems. For instance, showing keywords and parameters in example problems helps them to understand the situation and solution of the problems.

5. Conclusion

We proposed a method for problem-posing to develop problem solving ability in programming and developed a system for assisting to operate the method in actual classes. We also introduced the proposed method and the system into an actual fundamental programming class. The experimental result suggested the effect of the proposed method; however there are some problems that the number of the participants was insufficient, some of the participants did not understand the intention of the proposed method, and there were some participants, who could not pose problems though they had an example problem.

In this experiment, almost all of the participants understood the diversity of the situation and solution after the second explanation. However, it is better to give the student, who made wrong evaluation, feedback. To achieve it we are considering the method to determine the situational diversity of posed problem by comparing verbs used in posed problem and the example.

To confirm the effect of the proposed method we will continue to introduce the method and the system into actual class to collect data, improve instruction to inform the intention, and consider methods to support problem-posing tasks.

References

1. Y. Fuwa, H. Kunimune, M. Kayama, M. Niimura, H. Miyao, Implementation and evaluation of education for developing algorithmic thinking for students in computer science, IEICE Technical Report 109 (268) (2009) 51–56.
2. C. Kelleher, R. Pausch, Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers, ACM Computing Surveys 109 (2) (2005) 83–137.
3. A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, J. Paterson, A survey of literature on the teaching of introductory programming, in: Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education, ITiCSE-WGR '07, ACM, New York, NY, USA, 2007, pp. 204–223. doi:10.1145/1345443.1345441.
4. J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, M. Resnick, Scratch: A sneak preview, in: Proceedings of the Second International Conference on Creating, Connecting and Collaborating Through Computing, C5 '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 104–109. doi:10.1109/C5.2004.33.
5. T. Hirashima, T. Yokoyama, M. Okamoto, A. Takeuchi, Learning by problem-posing as sentence-integration and experimental use, in: Proceedings of the 2007 Conference on Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work, IOS Press, Amsterdam, The Netherlands, The Netherlands, 2007, pp. 254–261.
6. K. Kojima, K. Miwa, A system that facilitates diverse thinking in problem posing, International Journal of Artificial Intelligence in Education 18 (3) (2008) 209–236.
7. Y. Hirai, A. Hazeyama, A learning support system based on question-posing and its evaluation, in: Proceedings of the Fifth International Conference on Creating, Connecting and Collaborating Through Computing, C5 '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 178–184. doi:10.1109/C5.2007.2.
8. F. Y. Yu, Y. H. Liu, T. W. Chan, A web-based learning system for question-posing and peer assessment, Innovations in Education and Teaching International 42 (4) (2005) 337–348.
9. T. Hirashima, A model of learning by problem-posing, in: Proceedings of the 23rd Annual Conference of the Japanese Society for Artificial Intelligence, 2002, pp. 1–3.
10. J. P. Mestre, Probing adults' conceptual understanding and transfer of learning via problem posing, Journal of Applied Developmental Psychology 23 (1) (2002) 9–50. doi:http://dx.doi.org/10.1016/S0193-3973(01)00101-0.
11. A. Renkl, R. K. Atkinson, Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective, Educational Psychologist 38 (1) (2003) 15–22.
12. S. Shimada, H. Kunimune, M. Niimura, Proposal of a method of problem-posing and a system that assists to manage problems and assist peer evaluation., in: G. A. Tsihrintzis, M. Virvou, T. Watanabe, L. C. Jain, R. J. Howlett (Eds.), IIMSS, Vol. 254 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2013, pp. 128–137.
13. Y. Fujihara, H. Ohnishi, H. Kato, An evaluation support system for impartial peer evaluation : "reciprocal effect" occurred by selecting evaluators in case of peer evaluation of learning outcome, Journal of Japan Society for Educational Technology 31 (2) (2007) 125–134.